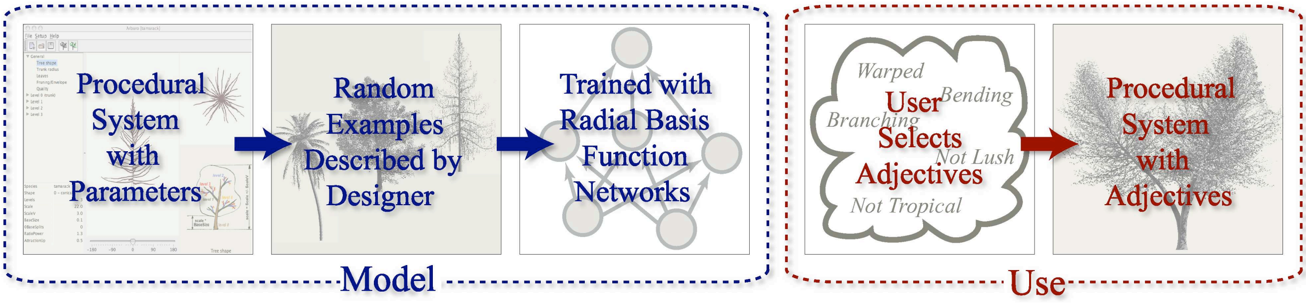
An Adjectival Interface for Procedural Content Generation

Carl Hultquist
chultquist@gmail.com
University of Cape Town

James Gain
jgain@cs.uct.ac.za
University of Cape Town

David Cairns
dec@cs.stir.ac.uk
University of Stirling



Abstract

This paper presents a novel interface for generating procedural models, textures, and other content, motivated by the need for interfaces that are simpler to understand and more rapidly utilize. Instead of directly manipulating procedural parameters, users specify adjectives that describe the content to be generated. By making use of a training corpus and semantic information from the WordNet database, our system is able to map from the set of all possible descriptions, *adjective space*, to the set of all combinations of procedural parameters, *parameter space*. This is achieved through a modification to radial basis function networks, and the application of particle swarm optimization to search for suitable solutions. By testing with three very different procedural generation systems, we demonstrate the wide applicability of this approach. Our results show that non-technical users not only prefer an adjectival interface to one offering direct control over the procedural parameters, but also produce content that more closely matches a given target.

Keywords: modeling, procedural methods, radial basis function networks, adjectives

1 Introduction

In today's world of fast-paced technological growth, the performance of CPUs and GPUs is increasing at an incredible rate. As such, the modern home computer is becoming much more capable of presenting ever larger and more complex digital content — for example, virtual environments, which are used extensively in computer games and simulations. There has been a corresponding demand to leverage these advances, by creating larger and more complex content and thus pushing the limits of the hardware available.

Fortunately, *procedural methods* [6] have come to the rescue, providing a means for machines to perform the mundane and repetitive aspects of digital content generation, and freeing up human resources for more interesting tasks. In essence, procedural methods are any form of procedure that takes some input — usually quite simple in nature — and transforms that input into complex content through a mechanical process. Whilst there are various interfaces to procedural methods (such as via sketches or image maps), most methods employ a *parametrised* interface for some aspect of control. Some methods rely solely on a parametrised interface — for example, the tree generation technique of Weber and Penn [21], which employs 80 parameters to control the generation of a single tree.

Whilst parametrised procedural models do well to abstract away the complexity of content generation, they do not necessarily provide a useful interface to all users. The nature of the parameters is typically such that a knowledge of the underlying procedure is required, in order to fully understand how the parameters affect the resulting output. As such, long training periods are typically required for users — such as visual effects artists — to become fully conversant with modern modeling systems.

Whilst striving to provide a simpler interface, it would also be prudent to maintain the fine degree of control afforded by procedural models to advanced users. As such, care must be taken not to over-simplify the interface and in so doing limit the power and flexibility of the procedural models.

In this paper, we present a technique that allows the user to generate procedural content using *adjectival descriptors*. This is achieved as an additional layer of abstraction that establishes a mapping between adjectives and the underlying procedural parameters, and thus addresses the issues raised thus far by providing the following features:

Allows large and complex procedural content to be created quickly. Existing procedural models are employed "under the hood", and these already provide for quick generation of complex output.

Provides a simpler interface, that is also usable by novice and non-technical users. All people communicate with language, using adjectives to guide their descriptions of objects and occurrences. An adjectival interface should thus be readily usable by both technical and non-technical users.

Maintains the flexibility afforded by parametrised procedural models. As the technique presented is implemented as an abstract layer on top of the procedural parameters of the model, it provides a form of learning support — known as *intrinsic scaffolding* — in that the adjectival interface can be used for initial generation of content, and further minor modifications can be made afterwards by the user at the procedural parameter level. The adjectival interface can also be seen as an intermediate bridging method, providing an easy means for a user to generate content until they are fully conversant with the underlying procedural parameters.

Having motivated and briefly introduced our technique, we now examine prior work in interfaces to procedural models, and in the use of adjectival descriptors. In Section 3 we discuss the details of our technique, which is followed in Section 4 by a discussion of our testing and the results obtained. Section 5 draws final conclusions, and presents ideas for future work in this area.

2 Related work

As our technique presents an adjectival interface to the generation of procedural content, we focus our attention on related work in the fields of interfaces to procedural content, and on the prior use of adjectival descriptions. We assume that the reader is conversant with the general concept of procedural modeling and its applications.

2.1 Interfaces to procedural modeling

Somewhat paradoxically, the automation motivating the use of procedural models is also a weakness, as the manipulation of procedural parameters offers less control than direct manipulation of the final output. As such, various interfaces to procedural modeling are used in an attempt to regain greater control.

Image maps [15] offer one means for improving usability and control, as they exploit most users' familiarity with image manipulation software. Image maps have been used extensively for a variety of purposes, such as land usage data for city generation [14], controlling feature placement for terrain synthesis [22], and for specifying distribution and density of plants in outdoor scenes [5].

Many complex objects exhibit shapes that are not easily captured via mechanical rules or inferences. Human designers are often far more adept at capturing such shapes through the use of *sketching*, and these sketches can be used to guide particular procedural modeling techniques. Successful uses of sketching include, amongst others, the procedural modeling of trees [11], motion [18], clothing [19] and terrain [3].

2.2 Use of adjectival descriptors and natural language

The use of textual descriptions in tagging media for later synthetic constructions has been explored extensively. One area which has received much attention is the creation of *stylized character motion* [16, 2]. A common paradigm for the representation of these stylistic features is to assign adverb descriptors axes in a multidimensional space, known as *adverb space* and coined by Rose et al. [16]. Given a point in adverb space and an action, the problem is then to produce a corresponding motion that takes on the adverb characteristics specified.

Using verbal descriptors in a different context, Barnard and Forsyth [1] present a method for hierarchically organizing a dataset of images by combining the semantic information of word-tags associated with each picture, with visual information given by features extracted from the images. Natural language has also been employed to describe the relationships between objects, for example in the system WordsEye [4] — an automatic text-to-scene conversion system that decomposes a piece of text into a dependency hierarchy, and uses the object names to index a database of models.

Hultquist et al. [9] make the first step of applying adjectival descriptors to parametrised procedural content, by proposing an adjectival interface for the creation of procedural virtual environments. Similar to the adverb space of Rose et. al., they define *adjective space* as the set of possible descriptions of an environment. They then posit the core problem as one of function approximation, and the need to find a function that maps from adjective space to parameter space. Using a *radial basis function network* (RBFN) [12], they show how this technique could be applied to a simple procedural landscape controlled by 16 parameters, and utilizing 7 adjectives.

2.3 Contributions made by our technique

Similarly to Hultquist et. al., we use adjectival descriptors to establish a simpler interface to parametrised procedural content. As such, our approach is very general and can be applied to a wide variety of procedural models, unlike the other interfaces discussed (sketching and image maps), which need to be configured specially for each procedural method. Our technique, however, differs significantly from that of Hultquist et. al. — through a different mapping and optimization scheme we overcome the deficiencies in their technique, and using an extension to RBFNs combined with semantic information from WordNet [7], our technique allows for the use of adjectival descriptors not necessarily tied to an axis of adjective space.

3 Implementation

We suppose that the user of our system will describe the content they wish to generate using a number of adjectival descriptors chosen from a set $\mathbf{A}$. Each adjective is tied to a dimension of adjective space, given by $\mathcal{A} = [-1; 1]^{|\mathbf{A}|}$. The value x in any dimension of $\mathcal{A}$ is the *scalar value* associated with a descriptor, and indicates the extent to which that descriptor applies — -1 denotes a definite absence of the adjectival descriptor, whilst 1 indicates a definite presence. We denote the set of procedural parameters by $\mathbf{P}$, and parameter space $\mathcal{P}$ as the subspace of $\mathbb{R}^{|\mathbf{P}|}$ in which each dimension is restricted to the range of real values spanned by the corresponding parameter.

To map between adjective space and parameter space, a number of points in parameter space are randomly chosen and fed through the procedural system to generate content. An expert user or artist then assigns adjectival descriptors to each piece of content, effectively giving pairs of corresponding points in adjective space and parameter space. By employing function approximation techniques on this training data, a mapping between the two spaces can be established. In our implementation, we employ RBFNs [12] as these are well established and also afford the incorporation of a useful extension, discussed in Section 3.3. Setting the parameters of RBFNs (such as basis function centers and their support radii) is known to be difficult, but we have found that making use of Orr's regression tree methods gives good results.

3.1 Mapping between adjective space and parameter space

In order to establish a mapping between adjective space and parameter space, we approximate the function $f : \mathcal{P} \rightarrow \mathcal{A}$. Whilst this may seem counter-intuitive as we seek a method for mapping from adjectives to procedural parameters, a solution exists in the form of *space-searching* techniques such as *particle swarm optimization* [10] or *genetic algorithms* [8]. If the user wishes to generate content with a description given by $\mathbf{a} \in \mathcal{A}$, then what is required is to solve $f(\mathbf{p}) = \mathbf{a}$. As f may not be onto, we relax this to instead solve $f(\mathbf{p}) \approx \mathbf{a}$ by minimizing the squared error

$$E(\mathbf{p}) \quad = \quad \|f(\mathbf{p}) - \mathbf{a}\|^2 \qquad (1)$$

In our implementation, we make use of the particle swarm optimization algorithm. The goal of the swarm of particles is to locate a point in parameter space that best matches the output in adjective space, using the error metric defined in Equation 1. The swarm will tend to move in the direction of the current best solution, but with a stochastic element which may lead it to find even better solutions as it converges.

Specifically, 2000 particles are assigned random positions with uniform probability, and initially have a velocity of $\mathbf{0}$. The particles' positions and velocities are then adjusted in an iterative fashion, drawing particles closer towards the locally and globally best observed positions whilst also applying stochastic perturbations. The algorithm terminates when either the error $E(\mathbf{p})$ of some particle $\mathbf{p}$ is less than 0.001, or when 15000 iterations have completed. As the algorithm is easily parallelized by dividing up the swarm, the running time of the algorithm can be kept below 30 seconds.

Our use of particle swarm optimization instead of a genetic algorithm is largely due to the fact that genetic algorithms are more intuitively suited to discrete problem domains, although there do exist means for genetic algorithms to be applied to real-valued domains. We have found that particle swarm optimization performs adequately; similar results could likely be achieved by using a genetic algorithm.

Using the inverse mapping affords a number of benefits:

Well-defined mapping. It is conceivable that two different pieces of generated content, with corresponding points $\mathbf{p_1}$ and $\mathbf{p_2}$ in parameter space, could have the same description given by $\mathbf{a} \in \mathcal{A}$. With our mapping this is perfectly acceptable, and since it is reasonable to suppose that any particular piece of content will be described by a user in a unique way, f is well defined. If instead one approximated the function $g = f^{-1} : \mathcal{A} \rightarrow \mathcal{P}$, it would be unclear what $g(\mathbf{a})$ should map to.

Interaction of procedural parameters. Multidimensional function outputs are typically addressed by approximating a separate function for each output dimension — as such, approximating g would lead to the procedural parameters being separated, which is undesirable in procedural models that exhibit interactions between their parameters. Our approximation of f overcomes this by instead separating the adjectival descriptors, and thus allowing for complex interactions and dependencies amongst the procedural parameters.

Reduced adjectival description burden. Since g maps from adjective space to parameter space, evaluating g requires the user to specify a value for *every* dimension of adjective space. Similarly, during training the expert user or artist would be required to specify a value for *every* dimension of adjective space for *every* piece of training content. For a large number of adjectives this can be quite a daunting proposition. Using f affords a more concise training by only requiring the expert user to specify adjectives which are pertinent to each individual piece of content. It also makes for easier content generation as users only need specify selected adjectival descriptors. If $\mathcal{I}$ is the set of dimension indices corresponding to descriptors chosen by the user, then this infers a modification of the sum-squared error from Equation 1 to give

$$E(\mathbf{p}, \mathcal{I}) = \sum_{i \in \mathcal{I}} [f(\mathbf{p})_i - a_i]^2$$

3.2 Dynamic use of additional adjectives

The presentation of adjective space thus far has made use of a fixed set of adjectival descriptors that the user is forced to use. Whilst this simplifies the problem and gives some degree of objectivity, it does confer an element of bias by suggesting to the user which descriptors they should use, as well as not allowing for other valid descriptors that may seem more natural to the user. The major difficulty in supporting new descriptors is in establishing relations to other known descriptors — if one were able to do this, then some degree of information could be inferred about the new descriptor so as to facilitate its usage.

We address this issue by using the WordNet database [7], which groups words into *synsets* — groups of words that in a given context have the same meaning — and also provides links between synsets that have similar and opposite meanings. To incorporate this information into the realm of adjective space, we associate with each training point an additional *certainty value*, k_i, that confers a measurement of the certainty of the observation. This can be used in conjunction with WordNet to amplify adjective space, by traversing the semantic relationship graph with decreasing certainty.

Formally, if content with procedural parameters $\mathbf{p}$ is described during the training process with descriptor $\mathcal{X}$ and associated scalar value x, then as the *root descriptor* this would be assigned a certainty value of $k_{\mathcal{X}} = 1$. If descriptor $\mathcal{Y}$ is similar to $\mathcal{X}$ according to the WordNet database, then we could also describe $\mathbf{p}$ with descriptor $\mathcal{Y}$ and scalar value x, but with a lower degree of certainty $k_{\mathcal{Y}} = d \cdot k_{\mathcal{X}}$, where $0 < d < 1$ is a value controlling the rate of decay in certainty. Antonymic relationships can be captured in a similar fashion, by associating the antonym with a scalar value of $-x$. This propagation of training data to related descriptors can then be repeated, either up to a certain number of levels from the root descriptor or until the certainty values fall below a predefined threshold.

Additionally, certainty values allow for further data amplification — if during training a particular descriptor is not used to describe some content, then this suggests that it does not apply to the content and so we could associate it with a scalar value of -1. It is possible, though, that the user simply overlooked the descriptor — certainty values come to the rescue, by assigning these inferred data points using a lower certainty value ($k = 0.2$ has worked well in our studies).

3.3 Incorporating certainty values into function approximation

As our technique requires the use of a space searching technique, a form of function approximation that provides rapid results is imperative in order to support the many computations performed during the search. RBFN's are one such candidate, and are also suitable for the incorporation of certainty values as will now be demonstrated.

A typical RBFN is a function of the form $f(\mathbf{x}) = \sum_{j=1}^{m} w_j h_j(\mathbf{x})$, where the h_j are the basis functions and the w_j are solved for by minimizing the cost function

$$C = \sum_{i=1}^{n} [f(\mathbf{x_i}) - y_i]^2 + \sum_{j=1}^{m} \lambda_j w_j^2 \qquad (2)$$

for n training pairs $(\mathbf{x_i}, y_i)$ and regularization parameters λ_j. Recall that certainty values, as the name implies, confer a measurement of the certainty of an observation. We would thus expect data that is less certain to have less impact on the approximant, and thus Equation 2 can be injected with certainty values to give

$$C = \sum_{i=1}^{n} k_i [f(\mathbf{x_i}) - y_i]^2 + \sum_{j=1}^{m} \lambda_j w_j^2$$

Following a derivation similar to that of a normal RBFN, this gives rise to a solution for the weights of

$$\mathbf{w} = \mathbf{A}^{-1}\mathbf{H}^\top\mathbf{K}\mathbf{y}$$

where

$$\mathbf{A} = \mathbf{H}^\top\mathbf{K}\mathbf{H} + \mathbf{\Lambda}, \qquad \mathbf{H} = \begin{pmatrix} h_1(\mathbf{p_1}) & \cdots & h_m(\mathbf{p_1}) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{p_n}) & \cdots & h_m(\mathbf{p_n}) \end{pmatrix}$$

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{pmatrix}, \qquad \mathbf{K} = \begin{pmatrix} k_1 & 0 & \cdots & 0 \\ 0 & k_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_n \end{pmatrix}$$

4 Testing and results

In order to evaluate the effectiveness of our technique, we conducted a user study in which the adjectival interface is compared and contrasted to an interface that offers direct manipulation of the numerical procedural parameters. A procedural model was created using Houdini [17], offering 49 procedural parameters that control the generation of an outdoor landscape. 500 points in the parameter space of this model were randomly chosen to train the RBFNs, and a set of 22 adjectival descriptors were used to describe the generated landscapes. Using WordNet to extrapolate to semantically connected synsets with a decay of $d = 0.7$, a total of 81 adjectival descriptors were made available to the user (see Appendix B). After the user chooses a subset of these adjectives, PSO finds a matching point in parameter space and the landscape is generated. The user is free to then modify their description and generate further landscapes until they are satisfied.

Objectively testing whether our interface provides any benefits over the direct manipulation of procedural parameters, is non-trivial. One could present the user with a piece of procedurally generated content, and have them use one of the interfaces to generate matching content, but in this case users may focus too much on minor matching details instead of considering a higher-level match. We address this concern through a two-stage experimental process.

In the first stage, users were shown a photograph of a real-world landscape, and were asked to create a virtual landscape that captured the spirit of the photograph as faithfully as possible. Each user was presented with either the adjectival[1] or direct manipulation interface, after first being given a 2 minute demonstration. To test for subject fatigue or learning bias, each user repeated the task with a second, different photograph. The users were limited to 22 minutes in which to perform each task, and once the

[1]Although the adjectival interface can be utilised as a form of scaffolding, in this study users of the adjectival interface were not permitted to "remove" the scaffolding and reveal the direct specification interface; they had to make sole use of the adjectival interface.

time had elapsed they were presented with all the landscapes that they had generated and were asked to select the best one. In this way, we avoid the possibility that users could focus on low-level matchings, due to the difference in realism between the photographs and the generated landscapes (see, for example, Figure 1). After completing the task, users completed a questionnaire relating to their experience — asking on a scale of 1 to 10 about how well they thought they had performed, their degree of frustration, the ease of use and understanding of the interface, and whether they could improve their performance with practice. This would provide quantitative data with which the two groups qualitative experience could be statistically compared.

The second stage of the user study made use of the final landscapes generated by users of the first stage. Participants in the second stage were presented with a photograph and two generated landscapes (one from each interface, unknown to the participant), and asked to choose which landscape more faithfully captured the photograph. This allows us to perform a blind and objective analysis of which interface produced more faithfully matching content.

In total, 5 photographs were used and randomly assigned to 35 first stage participants — 17 using the direct manipulation interface, and 18 using our adjectival interface. Appendix A shows the scores that users gave in response to the questionnaire (Tables 1, 2, 3 and 4), and Table 5 presents the results of t-tests used to compare the two user groups. This statistical analysis shows that users of the adjectival interface found the interface easier to use and understand, rated the matching of their generated environments more highly, and performed their task more quickly than users of the direct manipulation interface — all with a confidence level of greater than 95%.

In the second stage, 89 participants took part and each analyzed 15 sets of data, giving 1335 data points. Of these, 566 selected landscapes created using the direct manipulation interface, whilst 769 chose those created using the adjectival interface. To establish the statistical significance of this distribution, a binomial test gives a p-value of 1.522e-08 — well within the standard confidence interval of 95%. This indicates that the users' choices cannot have been made by a random process, and that users are statistically more likely to prefer landscapes generated using our adjectival interface, over those generated using the direct manipulation interface.

To show the applicability of our technique to other domains, we present some additional examples in the generation of trees [21]. Weber and Penn present a method for the procedural generation of a wide variety of tree types, focusing on the geometric structure of the tree as opposed to strictly adhering to botanical principles. They make use of 80 parameters which exhibit inherently complex interactions — for example, a parameter that controls the number of levels of branching, and which affects whether various parameters are used at all. Figure 2 shows some examples of how adjectival descriptors map to generated trees.

We have also applied our technique to the technique of Oudeyer [13], who describes an algorithm for the generation of meaningless baby-like speech that is able to impart various emotions, and which is controlled by 10 procedural parameters. Examples are available for download at `http://people.cs.uct.ac.za/~chultqui/speech_samples/`.

5 Conclusions and future work

In conclusion, we have presented a more natural approach to the generation of parametrised procedural content, by using adjectival descriptors. As an additional layer above procedural parameters, our approach does not replace existing procedural techniques but augments them with an alternative interface, providing scaffolding until a user is fully conversant with the model. User experiments have shown that novice users not only prefer this technique, but also that it results in content which more accurately matches the user's intentions. Finally, we have shown how this interface can be applied to various different procedural models.

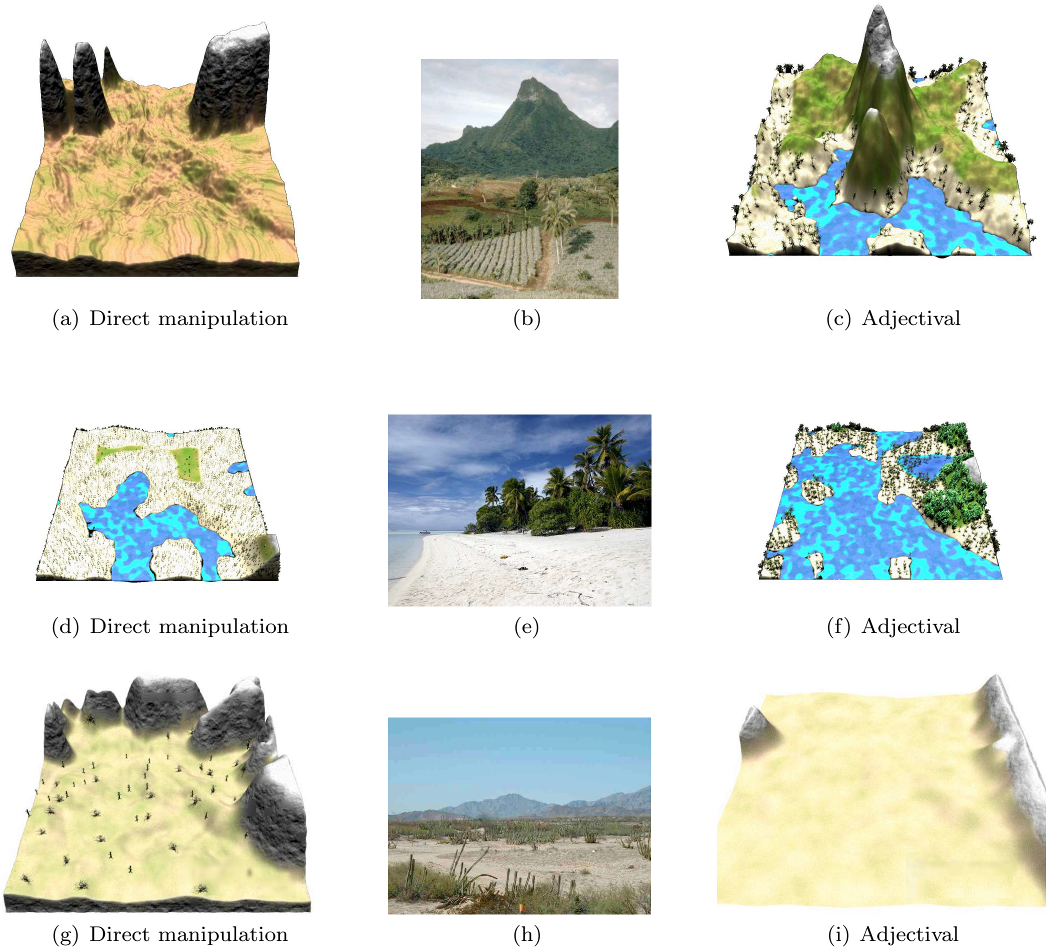
Our approach is not without caveats, however, and there is room for further improvements and extensions, such as:

Improved learning mechanism. Currently, the onus of training the RBFNs is on a single designer, or possibly a small group of designers who reach a consensus on descriptions for training data. This can be a large amount of work, and may make the use of this technique infeasible in some cases. The opinions of one designer may also not be well matched to the average opinion of the public as a whole, in which case even a well trained RBFN may not achieve adequate results for the average user. One means of addressing these issues might be through the use of a more widespread data collection process, with a suitable means for identifying outliers and normalizing the data. Certainty values may be of use here to weight data based on its trustworthiness.

Improved space-searching technique. Whilst we have achieved positive results, a potential bottle-neck in the process was the particle swarm optimization. To achieve fast optimization we utilized several network-linked machines; running on a single machine would have taken much longer to complete the optimization step, and would have severely impacted the interactivity of the task. It is possible that an improved learning mechanism might help by providing functions that are more easily optimized. In general, however, it would be interesting to further explore this optimization step by assessing the impact of different starting conditions, and also additional optimization algorithms.

Per-user training. Hultquist et al. correctly note that users express themselves in different ways, and that the function learned for one user may therefore not adequately map the perceptions of another user. Whilst we have not explicitly dealt with this issue — due, in part, to our achieving positive results without the need for this support — one way in which this could be approached is again through the use of certainty values. By augmenting the training data with a small number of additional examples that are provided on a per-user basis, and by assigning these greater certainty values, the function will more closely approximate the data provided by each user but will still use the common training corpus as a guide to the overall function approximation.

Exploration of other tools for dealing with higher dimensions. The curse of dimensionality means that discrete sampling becomes increasingly futile in higher dimensions. Whilst our testing has not suffered from this, some specific problem domains may require the use of some more complicated techniques. Methods such as principal components analysis or the use of latent variables may be useful in reducing the dimensionality of the space before applying our technique, or the use of newer function approximation methods that are specifically geared towards higher dimensions (such as that of Vijayakumar [20]) may be of better benefit than our RBFN implementation.



(a) Direct manipulation (b) (c) Adjectival

(d) Direct manipulation (e) (f) Adjectival

(g) Direct manipulation (h) (i) Adjectival

Figure 1: Some examples of our adjectival technique in comparison to direct manipulation of procedural parameters. (a) and (c) show user responses to the photo in (b); (d) and (f) in response to the photo in (e); (g) and (i) in response to the photo in (h). (c) was described as *volcanic, tropical, sandy, rocky* and not *flat*; (f) was described as *coastal, fragmented, sandy, tropical* and not *craggy*; (i) was described as *sunbaked,* not *flooded, sandy* and *lush.* As can be seen in (i), the adjectival interface does have some limitations and does not always correctly map the intentions of the user — in this case, the resulting content clearly is not lush. This could be because the user's perceptions differ from those of the expert user who trained the system, or could indicate that greater sampling of parameter space is required during the training phase.

6 Acknowledgements

We would like to thank the Harry Crossley Foundation, the Postgraduate Funding Office at the University of Cape Town, and the National Research Foundation, without whose support this research would not have been possible.

References

[1] K. Barnard and D. Forsyth. Learning the semantics of words and pictures. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 408–415, 2001.

[2] Matthew Brand and Aaron Hertzmann. Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] Jonathan M. Cohen, John F. Hughes, and Robert C. Zeleznik. Harold: a world made of drawings. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 83–90, New York, NY, USA, 2000. ACM.

[4] Bob Coyne and Richard Sproat. Wordseye: an automatic text-to-scene conversion system. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496, New York, NY, USA, 2001. ACM Press.

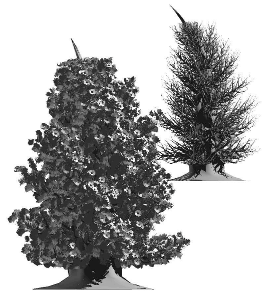
[5] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomr Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH 98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 275–286, New York, NY, USA, 1998. ACM Press.

[6] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc, 3 edition, 2003.

[7] C. Fellbaum. *WordNet – An Electronic Lexical Database*. MIT Press, 1998.

[8] David Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison Wesley, 1989.

[9] Carl Hultquist, James Gain, and David Cairns. Affective scene generation. In *Afrigaph '06: Proceed-*



(a) Described as *wide, lush, tapered, top-heavy,* not *skeletal,* not *drooping.*

(b) Described as *bending, branching, warped,* not *lush,* not *tropical* and not *majestic.*

Figure 2: Some examples of trees generated using our technique when applied to the work of Weber and Penn [21]. Whilst our technique produced adequate results, they do not always follow the adjectives as well as in our landscape experiment (see (b), for example). We believe this to be as a result of the much larger parameter space requiring a greater sampling, and discuss ways in which this could be dealt with in Section 5).

ings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, pages 59–63, New York, NY, USA, 2006. ACM Press.

[10] James Kennedy and Russell C. Eberhart. *Swarm intelligence.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[11] Makoto Okabe and Takeo Igarashi. 3d modeling of trees from freehand sketches. In *Proceedings of the SIGGRAPH 2003 conference on Sketches & applications*, pages 1–1, New York, NY, USA, 2003. ACM Press.

[12] Mark J. L. Orr. Recent advances in radial basis function networks. Technical report, Institute for Adaptive and Neural Computation, Division of Informatics, Edinburgh University, June 1999.

[13] P-Y. Oudeyer. The production and recognition of emotions in speech: features and algorithms. *International Journal of Human Computer Interaction*, 59(1-2):157–183, 2003. special issue on Affective Computing.

[14] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308. ACM Press, 2001.

[15] Ken Perlin and Luiz Velho. Live paint: painting with procedural multiscale textures. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 153–160, New York, NY, USA, 1995. ACM.

[16] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–41, 1998.

[17] SideFX. Houdini. `http://www.sidefx.com`, 2007.

[18] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 424–431, New York, NY, USA, 2004. ACM.

[19] Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John Hughes. A sketch-based interface for clothing virtual characters. *IEEE Computer Graphics & Applications*, January/February 2007.

[20] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Comput.*, 17(12):2602–2634, 2005.

[21] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *SIGGRAPH 95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1995. ACM Press.

[22] Howard Zhou, Jie Sun, Greg Turk, and James M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, July/August 2007.

A Questionnaire data

	Accuracy 1	Accuracy 2	Easy to under-stand	Easy to use	Frustration	Expectations met	Combined score
	6	4	7	7	6	4	22
	8	5	4	10	8	6	25
	1	5	3	7	3	5	18
	5	3	2	3	7	4	10
	1	3	1	2	6	3	4
	6	4	7	7	1	5	28
	5	5	2	3	6	3	12
	6	7	6	9	4	4	28
	6	5	5	6	2	3	23
	3	6	3	8	8	2	14
	2	4	2	3	8	2	5
	3	2	3	10	3	1	16
	3	5	3	3	5	5	14
	4	5	6	8	6	5	22
	1	5	1	2	7	3	5
	6	5	4	7	6	5	21
	6	8	6	10	4	6	32
Mean	4.24	4.76	3.82	6.18	5.29	3.88	17.59
Std deviation	2.17	1.44	2.01	2.92	2.14	1.45	8.58

Table 1: Responses from users of the direct specification interface in the first stage.

	Accuracy 1	Accuracy 2	Easy to under-stand	Easy to use	Frustration	Expectations met	Combined score
	9	4	5	6	7	5	22
	4	6	8	8	6	4	24
	6	8	9	10	6	4	31
	6	7	6	8	6	3	24
	6	4	9	10	6	3	26
	6	6	10	10	5	3	30
	7	5	10	10	7	5	30
	5	4	6	6	7	3	17
	3	6	8	7	9	4	19
	4	5	10	8	4	4	27
	4	8	10	10	6	2	28
	7	3	10	10	4	4	30
	5	7	10	10	8	4	28
	6	8	9	10	7	6	32
	4	7	10	8	5	6	30
	4	5	9	10	5	4	27
	7	7	10	9	3	4	34
	6	3	6	6	5	4	20
Mean	5.5	5.72	8.61	8.67	5.89	4	26.61
Std deviation	1.5	1.67	1.72	1.57	1.49	1.03	4.74

Table 2: Responses from users of the adjectival interface in the first stage.

	Additional time needed (minutes)	Expected time needed per photograph after practise, (minutes)
	10	20
	15	10
	5	7
	20	15
	60	15
	0	7
	60	30
	0	7
	10	10
	30	10
	60	12
	30	30
	0	10
	20	10
	20	40
	0	10
	20	5
Mean	21.18	14.59
Std deviation	20.96	9.84

Table 3: Responses from users of the direct specification interface in the first stage, on how much additional time they required per photograph and the expected time that they would need to spend on each photograph after sufficient practise.

	Additional time needed (minutes)	Expected time needed per photograph after practise, (minutes)
	0	10
	0	22
	0	15
	0	10
	0	15
	10	10
	0	10
	10	10
	0	12
	0	5
	0	5
	10	10
	15	10
	10	18
	5	15
	10	10
	0	15
	0	16
Mean	3.89	12.11
Std deviation	5.3	4.32

Table 4: Responses from users of the adjectival interface in the first stage, on how much additional time they required per photograph and the expected time that they would need to spend on each photograph after sufficient practise.

Null hypothesis	t	df	p
$\mu_{\text{ADJ(accuracy 1)}} \leq \mu_{\text{DS(accuracy 1)}}$	1.9953	28.366	0.02785
$\mu_{\text{ADJ(accuracy 2)}} \leq \mu_{\text{DS(accuracy 2)}}$	1.8189	32.719	0.03905
$\mu_{\text{ADJ(easy to understand)}} \leq \mu_{\text{DS(easy to understand)}}$	7.5573	31.586	7.154e-09
$\mu_{\text{ADJ(easy to use)}} \leq \mu_{\text{DS(easy to use)}}$	3.1152	24.244	0.002338
$\mu_{\text{ADJ(frustration)}} \geq \mu_{\text{DS(frustration)}}$	0.9478	28.38	0.8244
$\mu_{\text{ADJ(expectations met)}} \leq \mu_{\text{DS(expectations met)}}$	0.275	28.693	0.3926
$\mu_{\text{ADJ(combined score)}} \leq \mu_{\text{DS(combined score)}}$	3.8195	24.632	0.000401
$\mu_{\text{ADJ(extra time)}} \leq \mu_{\text{DS(extra time)}}$	3.303	17.931	0.001986
$\mu_{\text{ADJ(expected time)}} \leq \mu_{\text{DS(expected time)}}$	0.9549	21.692	0.1751

Table 5: T-test results comparing the scaled data in Tables 1, 2, 3 and 4. $\mu_{\text{ADJ(x)}}$ indicates the mean of column x in the adjectival interface data; $\mu_{\text{DS(x)}}$ indicates the mean of column x in the direct specification interface data. The t, df and p columns give the t-value, degrees of freedom and p-value of the test, respectively.

B Adjectival descriptors provided for landscape experiment

dry	wet	even
full	dried	heavy
humid	misty	steep
tacky	inland	rheumy
sloppy	sticky	washed
watery	air-dry	coastal
covered	divided	gradual
inshore	seaward	thirsty
undried	abundant	besprent
detached	dried-up	dry-shod
maritime	rainless	semi-dry
semiarid	bone-dry	volcanic
air-dried	steepish	overgrown
patterned	coastwise	kiln-dried
landlocked	equatorial	distributed
sparse, thin	clammy, dank	bedewed, dewy
inhospitable	steep-sided	perpendicular
proportionate	sodden, soppy	arid, waterless
drippy, drizzly	showery, rainy	rough, unsmooth
steaming, steamy	reeking, watery	bluff, bold, sheer
flat, level, plane	tropical, tropic	damp, dampish, moist
argillaceous, clayey	muggy, steamy, sticky	desiccated, dried-out
freestanding, separate	abrupt, precipitous, sharp	arenaceous, sandy, sandlike
interior, midland, upcountry	rocky, bouldery, bouldered, stony	bare, barren, bleak, desolate, stark
cragged, craggy, hilly, mountainous	disproportionate, disproportional	adust, baked, parched, scorched, sunbaked
disconnected, disunited, fragmented, split	dotted, flecked, specked, speckled, stippled	exuberant, lush, luxuriant, profuse, riotous
afloat, awash, flooded, inundated, overflowing	dried-up, sere, sear, shriveled, shrivelled, withered	boggy, marshy, miry, mucky, muddy, quaggy, sloppy, sloughy, soggy, squashy, swampy, waterlogged

Table 6: A list of the descriptors provided to users of the adjectival interface in the first stage of the landscape experiment. Each entry in the table corresponds to a synset from the WordNet database, and lists the adjectives comprising that synset.